Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently Amended) A method in a data processing system for processing a JAVASERVER page, the method comprising:

translating the JAVASERVER page into a document object model object, the document object model object including a set of nodes;

configuring a set of visitor classes for invocation in a selected sequence;

processing the document object model using the set of visitor classes in the selected sequence to perform a desired set of custom functions on the document object model, wherein the processing step includes invoking methods in the set of visitor classes on each node in the set of nodes in the selected sequence; and

storing results, as processing the document object model object occurs, by selected method in the methods, in a hash map, wherein the results in the hash map are used by subsequently invoked methods, wherein the subsequently invoked ~~method does~~ methods do not convert the results into a second document object model object.

2. (Previously Presented) The method of claim 1 further comprising:

validating syntax in the JAVASERVER page.

3. (Original) The method of claim 1, wherein the set of visitor classes for invocation in the selected sequence is defined in a configuration file.

4. (Original) The method of claim 3, wherein the configuration file is an extensible markup language file.

5. (Original) The method of claim 3, wherein the selected sequence is defined in the configuration file.

6. (Canceled)

7. (Canceled)

8. (Previously Presented) The method of claim 1, wherein the JAVASERVER page is translated into a document object model object using a document object model generator.

9. (Previously Presented) The method of claim 2, wherein the JAVASERVER page is validated using a JAVASERVER page translator.

10. (Previously Presented) The method of claim 9, wherein the JAVASERVER page translator invokes a visitor class to validate elements in the document object model object against a syntax for a JAVASERVER page specification.

11. (Previously Presented) The method of claim 1 wherein results from processing using a first visitor class in the set of visitor classes are passed to a second visitor class in the set of visitor classes.

12. (Currently Amended) A data processing system for processing a JAVASERVER page, the data processing system comprising:

a memory containing a set of instructions; and

a processing unit, responsive to execution of the set of instructions, for translating the JAVASERVER page into a document object model object, the document object model object including a set of nodes; for configuring a set of visitor classes for invocation in a selected sequence; for processing the document object model using the set of visitor classes in the selected sequence to perform a desired set of custom functions on the document object model, wherein the processing step includes invoking methods in the set of visitor classes on each node in the set of nodes in the selected sequence; and for storing results, as processing the document object model object occurs by selected method in the methods, in a hash map, wherein the results in the hash map are used by subsequently invoked methods.

13. (Previously Presented) The data processing system of claim 12 wherein the processing unit, responsive to execution of the set of instructions, further comprises validating syntax in the JAVASERVER page.

14. (Original) The data processing system of claim 12, wherein the set of visitor classes for invocation in the selected sequence is defined in a configuration file.

15. (Original) The data processing system of claim 14, wherein the configuration file is an extensible markup language file.

16. (Original) The data processing system of claim 14, wherein the selected sequence is defined in the configuration file.

17. (Canceled)

18. (Canceled)

19. (Previously Presented) The data processing system of claim 12, wherein the JAVASERVER page is translated into a document object model object using a document object model generator.

20. (Previously Presented) The data processing system of claim 13, wherein the JAVASERVER page is validated using a JAVASERVER page translator.

21. (Previously Presented) The data processing system of claim 20, wherein the JAVASERVER page translator invokes a visitor class to validate elements in the document object model object against a syntax for a JAVASERVER page specification.

22. (Currently Amended) The data processing system of claim [[1]]12, wherein results from processing by a first visitor class in the set of visitor classes are passed to a second visitor class in the set of visitor classes.

23. (Currently Amended) A computer program product in recordable-type medium for processing a JAVASERVER page, the computer program product comprising:

first instructions for translating the JAVASERVER page into a document object model object, the document object model object including a set of nodes;

second instructions for configuring a set of visitor classes for invocation in a selected sequence;

third instructions for processing the document object model using the set of visitor classes in the selected sequence to perform a desired set of custom functions on the document object model, wherein the processing step includes invoking methods in the set of visitor classes on each node in the set of nodes in the selected sequence; and

fourth instructions for storing results, as processing the document object model object occurs by selected method in the methods, in a hash map, wherein the results in the hash map are used by subsequently invoked methods.

24. (Previously Presented) The computer program product of claim 23 further comprising: fourth instructions for validating syntax in the JAVASERVER page.

25. (Original) The computer program product of claim 23, wherein the set of visitor classes for invocation in the selected sequence is defined in a configuration file.

26. (Original) The computer program product of claim 25, wherein the configuration file is an extensible markup language file.

27. (Original) The computer program product of claim 25, wherein the selected sequence is defined in the configuration file.

28. (Canceled)

29. (Canceled)

30. (Previously Presented) The computer program product of claim 23, wherein the JAVASERVER page is translated into a document object model object using a document object model generator.

31. (Previously Presented) The computer program product of claim 24, wherein the JAVASERVER page is validated using a JAVASERVER page translator.

32. (Previously Presented) The computer program product of claim 31, wherein the JAVASERVER page translator invokes a visitor class to validate elements in the document object model object against a syntax for a JAVASERVER page specification.

33. (Original) The computer program product of claim 23, wherein results from processing by a first visitor class in the set of visitor classes are passed to a second visitor class in the set of visitor classes.

34. (Currently Amended) A data processing system for processing a JAVASERVER page, the data processing system comprising:

a bus system;

a memory connected to the bus system, wherein the memory includes a set of instructions;

a processing unit connected to the bus system, wherein the processing unit executes the set of instructions to translate the JAVASERVER page into a document object model object, the document object model object including a set of nodes; configure a set of visitor classes for invocation in a selected sequence; process the document object model using the set of visitor classes in the selected sequence to perform a desired set of custom functions on the document object model, wherein the processing step includes invoking methods in the set of visitor classes on each node in the set of nodes in the selected sequence; and storing results, as processing the document object model object occurs by selected method in the methods, in a hash map, wherein the results in the hash map are used by subsequently invoked methods.